

Sentinel Legacy Codeless Connector Platform

PROOF OF CONCEPT - STEP BY STEP GUIDE

THIS GUIDE SHOWS HOW TO CREATE A POC OF A SENTINEL LEGACY CODELESS CONNECTOR - USEFUL TO DEMONSTRATE IN YOUR PERSONAL ENVIRONMENT

Contents

INTRODUCTION 3

ARCHITECTURE 5

CONNECTOR 7

REFERENCE 11

INTRODUCTION

It is possible to collect data in Sentinel using agent based solutions (Microsoft Monitoring Agent, Azure Monitor Agent) or agentless (Diagnostic Settings, Sentinel Rest APIs – and therefore Function Apps, Logic App, any script that allows API calls).

This is independent against the Sentinel Data Connector concept. The Data Connector is simply a tool that makes it easier for us to collect data – sometimes giving step-by-step guides, other times completely hiding the logic for collecting data by showing just a single 'connect' button.

Example

Entra ID Connector: you select the data of interest and with a simple 'Active' button you're done. After less than 20 minutes, the data is available below in Sentinel.

(Spoiler: Entra ID Connector is based on Diagnostic Settings. After activating it you will find a new rule in Entra > Diagnostic Settings > AzureSentinel_nameOfLAW.

Azure Activity Connector - is based on assigning a DeployIfNotExists policy.

Google Cloud Platform IAM (using Azure Functions) Connector - as the name suggests, with a simple click an Azure Function is deployed.

Data connectors are available in the Content Hub, Github or you can create your own custom connector – documentation here > [Resources for creating Microsoft Sentinel custom connectors](#) | [Microsoft Learn](#)

Method description	Capability	Serverless	Complexity
Codeless Connector Platform (CCP) Best for less technical audiences to create SaaS connectors using a configuration file instead of advanced development.	Supports all capabilities available with the code.	Yes	Low; simple, codeless development
Log Analytics Agent Best for collecting files from on-premises and IaaS sources	File collection only	No	Low
Logstash Best for on-premises and IaaS sources, any source for which a plugin is available, and organizations already familiar with Logstash	Available plugins, plus custom plugin, capabilities provide significant flexibility.	No; requires a VM or VM cluster to run	Low; supports many scenarios with plugins
Logic Apps High cost; avoid for high-volume data Best for low-volume cloud sources	Codeless programming allows for limited flexibility, without support for implementing algorithms. If no available action already supports your requirements, creating a custom action may add complexity.	Yes	Low; simple, codeless development
PowerShell Best for prototyping and periodic file uploads	Direct support for file collection. PowerShell can be used to collect more sources, but will require coding and configuring the script as a service.	No	Low
Log Analytics API Best for ISVs implementing integration, and for unique collection requirements	Supports all capabilities available with the code.	Depends on the implementation	High
Azure Functions Best for high-volume cloud sources, and for unique collection requirements	Supports all capabilities available with the code.	Yes	High; requires programming knowledge

But what is the Codeless Connector Platform (CCP)?

Let's have an example.

There is a third party service that we are interested in ingesting logs into Sentinel. The service exposes a public REST API endpoint. You can think of developing a Function App in Python, which runs once every 20 minutes and which connects to the REST API endpoint retrieving the data (i) and then ingesting it into Sentinel by calling the Log Analytics API (ii).

Three pain points:

1. You should have experience in coding – which could be minimized by using a Logic App but not eliminated
2. You need to maintain the infrastructure (in this case the health of the Function App)
3. Cost associated with Function App executions

The Codeless Connector Platform resolves the issues mentioned above.

Key benefits include (from [The Codeless Connector Platform - Microsoft Community Hub](#)):

- Avoid writing lines of code to connect with publicly exposed REST APIs
- Scalable built in Poller as a service
- Configurable UI components for your connector
- Ingest Cost benefits
- Monitor your connectors: CCP integrates with Sentinel Connector Health message using which you can troubleshoot and get health messages.

This guide focuses on the legacy version of CCP – the new version is in preview, out of the scope here.

Everything is based on the creation of a json file that defines the connector interface and the endpoint from which to retrieve the data. Nothing more. It uses the Sentinel backbone to perform Endpoint API calls.

ARCHITECTURE

There is a third party service that we are interested in ingesting logs into Sentinel. The service exposes a public POST REST API endpoint. The public endpoint returns a json response as which has the following structure:

```
{
  "status": "OK",
  "items": [
    {
      "name": "zgjzt",
      "surname": "xvzym"
    }
  ]
}
```

We want to store *name* and *surname* in the **TestData_CL** custom table.
To simulate the third-party service you can create a Function App – for simplicity in Python.
Some useful take-note during the creation process:

- Deploy Code and not Container Image
- Runtime stack Python – Version 3.11
- Hosting options and plans: Consumption
- Enable Public Access
- Template HTTP trigger
- Job type create new app
- Authorization level FUNCTION

The screenshot displays the Azure portal interface for a Function App named 'testguideagain'. The left sidebar shows navigation options like Overview, Activity log, Access control, and Functions. The main area shows the 'Create function' dialog with the following details:

- Development environment:** Develop in portal
- Programming Model:** v2 Programming Model
- Select a template:** HTTP trigger (selected)
- Template details:** We need more information to create the HTTP trigger function.
- Job type:** Create new app
- Provide a function name:** http_trigger
- Authorization level:** FUNCTION

The 'Create function' dialog also includes a 'Create in Azure portal' button and a 'VS Code' button.

The python function is as follows (also available [here](#))

```
import azure.functions as func
import logging
import json
import random
import string

app = func.FunctionApp(http_auth_level=func.AuthLevel.FUNCTION)
@app.route(route="http_trigger")

def http_trigger(req: func.HttpRequest) -> func.HttpResponse:
    logging.info('Python HTTP trigger function processed a request.')
    name = ''.join(random.choices(string.ascii_lowercase, k=5))
    surname = ''.join(random.choices(string.ascii_lowercase, k=5))
    return func.HttpResponse(
        #body=json.dumps([{"name": name, "surname": surname}]),
        body=json.dumps({
            "status": "OK",
            "items" : [{"name": name, "surname": surname}]
        }),
        status_code=200,
        mimetype="application/json"
    )
```



CONNECTOR

In this section we analyse the structure of the ARM template used to deploy the connector.

This is general structure

```
{
  ...
  "resources": [
    {
      ...
      "properties": {
        "connectorUiConfig": {
          },
        "pollingConfig": {
          }
        }
      }
    ]
  }
```

The two most important properties of the Json file are *connectorUiConfig* and *pollingConfig*: the first section contains the graphical interface of the connector (title, description, publisher, query examples, step-by-step description for deployment) and the second contains information on the how to recover data (endpoint, authentication, how often to make the API call, how long to wait in case of error).

The entire Json file is as follows (also available [here](#))

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "workspace": {
      "type": "string",
      "defaultValue": ""
    }
  },
  "resources": [
    {
      "id": "[concat('/subscriptions/',subscription().subscriptionId,'/resourceGroups/',resourceGroup().name,'/providers/Microsoft.OperationalInsights/workspaces/',parameters('workspace'),'/providers/Microsoft.SecurityInsights/dataConnectors/',guid(subscription().subscriptionId))]",
      "name": "[concat(parameters('workspace'),' /Microsoft.SecurityInsights/',guid(subscription().subscriptionId))]",
      "apiVersion": "2021-03-01-preview",
      "type": "Microsoft.OperationalInsights/workspaces/providers/dataConnectors",
      "kind": "APIPolling",
      "properties": {
        "connectorUiConfig": {
```

```

    "title": "Personal Legacy CodelessConnector",
    "id": "LegacyCodelessConnector",
    "publisher": "Mario Cuomo Dev",
    "descriptionMarkdown": "This connector is used to retrieve data from an
Azure Function <insert your API Endpoint>. Fetch every minute.",
    "graphQueriesTableName": "TestData_CL",
    "graphQueries": [
        {
            "metricName": "Total data received",
            "legend": "Audit Events",
            "baseQuery": "{{graphQueriesTableName}}"
        }
    ],
    "sampleQueries": [
        {
            "description": "Distinct name value",
            "query": "{{graphQueriesTableName}}\n | distinct name\n"
        }
    ],
    "dataTypes": [
        {
            "name": "{{graphQueriesTableName}}",
            "lastDataReceivedQuery": "{{graphQueriesTableName}}\n
summarize Time = max(TimeGenerated)\n | where isnotempty(Time)" |
        }
    ],
    "connectivityCriteria": [
        {
            "type": "SentinelKindsV2",
            "value": [
                "APIPolling"
            ]
        }
    ],
    "availability": {
        "status": 1,
        "isPreview": false
    },
    "permissions": {
        "resourceProvider": [
            {
                "provider": "Microsoft.OperationalInsights/workspaces",
                "permissionsDisplayText": "read and write permissions are required.",
                "providerDisplayName": "Workspace",
                "scope": "Workspace",
                "requiredPermissions": {
                    "action": true,
                    "write": true,
                    "read": true,
                    "delete": true
                }
            }
        ]
    },
    "instructionSteps": [
        {
            "title": "Authenticate against the Azure Function",
            "description": "Provide the Function API Key.",
            "instructions": [
                {

```

```

        "type": "APIKey"
      }
    ]
  },
  "pollingConfig": {
    "auth": {
      "authType": "APIKey",
      "APIKeyName": "x-functions-key",
      "IsAPIKeyInPostPayload": false
    },
    "request": {
      "apiEndpoint": "<insert your API Endpoint>",
      "rateLimitQPS": 2,
      "httpMethod": "Post",
      "queryTimeFormat": "yyyy-MM-ddTHH:mm:ssZ",
      "retryCount": 3,
      "queryWindowInMin": 1,
      "timeoutInSeconds": 20
    },
    "response": {
      "eventsJsonPaths": [
        "$.items"
      ],
      "format": "json",
      "successStatusJsonPath": "$.status",
      "successStatusValue": "OK"
    }
  }
}
]
}

```

Some useful take-note during the creation process



- L'apiEndpoint is like `https://<FunctionAppName.azurewebsites.net/api/FunctionName`
- Authentication is performed by passing the apikey for the *x-functions-key* in the header
- The data to be stored in the *TestData_CL* table (graphQueriesTableName property) is in the nested *items* object
- The connector is active by checking the *status* property of the object returned from the apiEndpoint

Once the connector is deployed, you can see it in Sentinel.

To connect, you need to provide the value of an available api key - even the default one.

Personal Legacy CodelessConnector ...

Personal Legacy CodelessConnector

Not connected  **Mario Cuomo Dev**  Last Log Received: 11/11/2023 12:00:00 PM

Description
This connector is used to retrieve data from an Azure Function every minute.

Last data received: --

Related content

- 0** Workbooks
- 1** Queries
- 0** Analytics rules templates

Data received [Go to log analytics](#)

100
80
60
40
20
0

11 March 13 March 15 March

Total data received: **0**

Data types
TestData_CL

Instructions

Prerequisites

To integrate with Personal Legacy CodelessConnector make sure you have:

- ✔ **Workspace:** read and write permissions are required.

 Configuration

Authenticate against the Azure Function
Provide the Function API Key.

API key

Key

Connect Disconnect

[Home](#) > [Function App](#) > [testguideagain](#) > [http_trigger](#)

http_trigger | Function Keys ...

[+ New function key](#)
[Refresh](#)
[Send us your feedback](#)

[Overview](#)

Developer

Code + Test

 Integration

Monitor

Function Keys

Function keys

Function keys are scoped to this function and can be used to access this function.

Search

Name	Value
default	<div> ⋮ Show value Renew key value </div>

Wait 30 minutes and you can start seeing data in Sentinel!

Run Time range: Last 24 hours Save Share + New alert rule Export Pin to

1 TestData CL

2

Results Chart |  Add bookmark

<input type="checkbox"/>	TimeGenerated [UTC] ↑↓	name_s	surname_s	Type
<input type="checkbox"/>	> 16/03/2024, 17:35:26.346	brzic	ospgp	TestData_CL
<input type="checkbox"/>	> 16/03/2024, 17:34:22.194	pyxgm	xcpov	TestData_CL
<input type="checkbox"/>	> 16/03/2024, 17:33:20.733	uediz	idpsh	TestData_CL
<input type="checkbox"/>	> 16/03/2024, 17:32:23.757	fixyto	obyzb	TestData_CL
<input type="checkbox"/>	> 16/03/2024, 17:31:27.404	wbaty	sdsqr	TestData_CL
<input type="checkbox"/>	> 16/03/2024, 17:30:21.503	ahgbh	zzgyb	TestData_CL
<input type="checkbox"/>	> 16/03/2024, 17:29:22.755	slpdn	pozru	TestData_CL
<input type="checkbox"/>	> 16/03/2024, 17:28:26.123	pkrvk	apaaz	TestData_CL

REFERENCE

Some useful links

- Codeless Connector Platform, techcommunity blog
[The Codeless Connector Platform - Microsoft Community Hub](#)
- Legacy Connector Platform Docs
[Legacy codeless connector for Microsoft Sentinel | Microsoft Learn](#)
- LastPass Connector – an example available in Azure-Sentinel official Github Repository
[Azure-Sentinel/Solutions/LastPass/Data Connectors/LastPassAPIConnector.json at master · Azure/Azure-Sentinel · GitHub](#)
- Github Repository
[mariocuomo/LegacyCodelessConnectorSentinel: Proof of concept for a Sentinel connector based on the Codeless Connector Platform \(github.com\)](#)
- Step-by-Step Youtube video
<https://youtu.be/g757IDY-8h4>